Database Issues for Intelligence Analysis

Robert Ayres, Ross D C Harris, Graham Lee, Steve J Smith
Department of Information Systems
Cranfield University, Royal Military College of Science
Shrivenham, Swindon SN6 8LA, UK

Email: R.Ayres@cranfield.ac.uk

SUMMARY

The use of database systems in police investigative work is well established and similar technology will be required to support counter terrorism. A key feature of terrorism is that it is predominantly a group activity. This has important implications for analysing terrorism-related intelligence in that systems must help analysts uncover patterns of activity which may be obscured by being distributed over a network of individuals or organisations. In this paper we argue that the record-oriented representation of data on which most current database technology is based is not well-suited for such applications. Another form of representation, known as graph-based databases, organises information as a network of connected facts and is better suited to supporting queries concerned with finding associations and links between various entities such as people, events, organisations and so forth. We briefly discuss a database system that has been built on top of a graph-oriented representation of data in order to show that such systems are feasible. Furthermore such systems can support a broader range of queries, in particular the kinds of queries that are likely to be required in intelligence analysis.

1.0 INTRODUCTION

The use of database systems to support police investigations is well established. As forces have become more expert with the technology so they have used techniques, such as crime profiling or analysing criminal networks, to exploit the information they hold in support of investigative work. Counter terrorism will make use of similar approaches and techniques. However, there are particular features of counter terrorism that pose additional challenges for database technology.

One aspect of counter terrorism is that it is an intelligence-led activity. In criminal investigations information about an incident or suspects is generally gathered and stored after the incident has occurred. With counter terrorism however this sequence is reversed and large quantities of data are likely to be gathered and analysed to support the process of tracking and predicting terrorist activity. Even data of no apparent importance is likely to be retained in case its significance becomes apparent at a later stage. As a result the diversity of information held is likely to be much greater than for conventional police investigations and it may not be possible in advance to predict all the kinds of information that one needs to hold.

Another important aspect is that terrorism is generally a group or organisational activity. Consequently the analysis of terrorist activities and threats cannot confine itself to consideration of particular individuals or to series of similar events. Rather it has to track entire groups and to consider the ways in which individuals, events and activities are interrelated. For example, Figure 1 shows the way in which a number of people are associated with each other, an import company, ZZ Imports, and attendance at an arms fair. Sub-portions of this network on their own are not significant as many people attend various trade fairs and there will always be people who are known to have affiliations with extremist groups.

Paper presented at the RTO SCI Symposium on "Systems, Concepts and Integration (SCI) Methods and Technologies for Defence Against Terrorism," held in London, United Kingdom, 25-27 October 2004, and published in RTO-MP-SCI-158.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **25 OCT 2004**	2. REPORT TYPE **N/A**	3. DATES COVERED **-**

4. TITLE AND SUBTITLE **Database Issues for Intelligence Analysis**	5a. CONTRACT NUMBER
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Department of Information Systems Cranfield University, Royal Military College of Science Shrivenham, Swindon SN6 8LA, UK**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR’S ACRONYM(S)
	11. SPONSOR/MONITOR’S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release, distribution unlimited

13. SUPPLEMENTARY NOTES

See also ADM201977, Systems, Concepts and Integration Methods and Technologies for Defence against Terrorism (Systemes, concepts, methodes d’integration et technologies pour la luttre contre le terrorisme)., The original document contains color images.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE	**UU**	**33**	
unclassified	**unclassified**	**unclassified**			

However these separate facts become much more significant when they are connected in the way shown. Techniques for understanding these kinds of patterns, such as link analysis or social network analysis, are increasingly used by police departments, especially where the investigation of organised crime is concerned [16, 21].

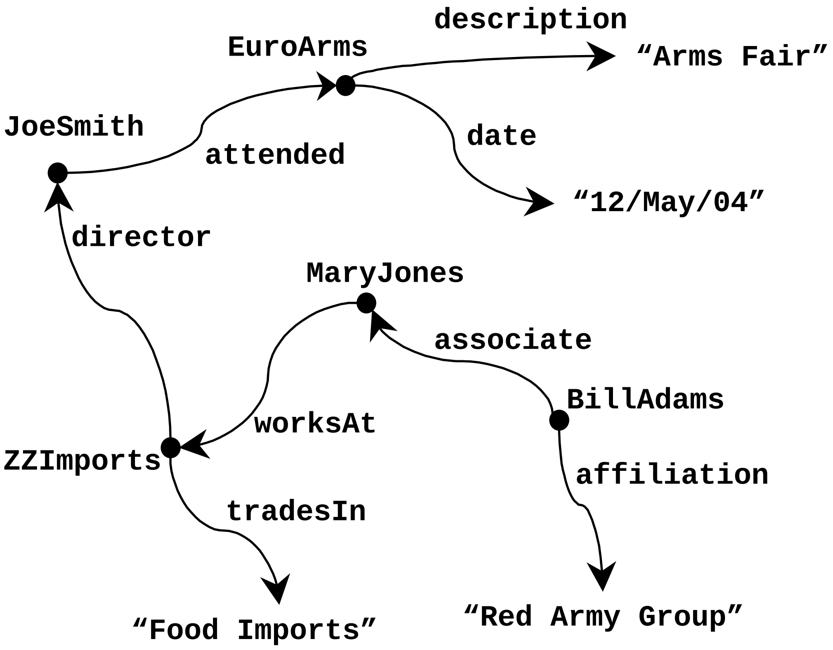


Figure 1: Associations between Three People

Coping with unforeseen kinds of data and searching for potential connections between disparate facts or elements are facilities that are not well supported by current technology. Most database systems are record-based, a record being essentially a template for holding a group of related facts. An employee database for example may use, among other record types, an Employee record that holds the name, employee number, salary, grade, and date of joining of the employee. For every employee represented in the database this same set of facts will be recorded. Most databases will store information using many different record formats according to the application the database is supporting.

The use of record-based databases has implications for intelligence analysis and storage. Firstly they require that the record formats to be used in the database be largely specified in advance giving rise to the need for a database to be designed before it is set up. Such design is complicated in situations where it is not possible to know all the types of information that may need to be held. Another issue, particularly important in the context of intelligence analysis, is that record-based databases tend to obscure the way that entities and facts may be interconnected. The record formats used by any normal application will be both diverse and relatively complex and it is difficult to develop general-purpose software that can explore the connectivity that is implicit in the data.

A consequence of using record-based database technology is that data often has to be reformatted in some way to allow a particular technique, such as link analysis or geographical profiling, to be applied. However once it has been reformatted the data in its new form can no longer be queried and analysed using standard database query facilities.

The ideal would be to hold intelligence data in a form that supports both database query features whilst also allowing a range of other techniques, such as link analysis, to be applied. One approach to providing a more flexible system would be to use database systems that are graph-based. By this we mean that the underlying data model used to organise information is no longer the record but simple associations between pairs of values. Such a data model represents information in a way which is directly analogous to the diagram in Figure 1. In effect the information becomes a network of individual items of information linked by associations. There are two important implications of this in the context of intelligence analysis: firstly it is much easier to build the database structure up incrementally by adding new kinds of association as required and, secondly, queries that are concerned with looking for associations can easily be supported. Record-based databases generally do not allow the user to execute a query that corresponds to the question "*What is the connection between Bill Adams and an arms fair?*" However with a database built over a graph data model such queries can be interpreted as searches for paths through the data and can, in principle, be supported alongside other standard queries.

This paper will briefly review one database system, Hydra, which has been built over a graph-based data model and was designed to support queries involving searches for associations. The particular advantage of Hydra is that it supports exploratory queries of the form "*What is known about X?*" or "*How are X and Y associated with each other?*" This additional power is a direct consequence of the underlying data model used and has not required sacrificing standard query facilities. The Hydra system can evaluate any query that can be posed using a conventional database system.

In the rest of the paper we shall first survey a range of analysis techniques that may be applied to intelligence data and discuss the problems that sometimes arise when using these techniques. One particular problem we shall discuss arises from the way in which data is modelled and the consequent requirement for pre-processing to allow a technique to be applied. The issue of data models and in particular the fundamental distinction between record-based models and so-called graph-based, or binary relational, models is explored before giving an overview of the Hydra system. In the conclusion a brief review of the possible ways in which graph-based data models can be exploited is given and some further issues that need to be addressed in the context of intelligence analysis are considered.

2.0 COMMON ANALYSIS TECHNIQUES

Law enforcement work has always required the management of large amounts of information in the form of case notes and reports. Information Technology (IT) has been utilised to support the storage, retrieval and analysis of this information through the use of record-based structures, often in relational database systems [7]. These databases can be interrogated using, for instance, Structured Query Language (SQL) statements to apply user-specified criteria for filtering the data [8]. The results of queries such as "*Who owns a red Ford Fiesta car and lives at an address with a Nottingham, UK area post code?*" can be used to support standard investigative work.

It is a logical step to move from using a database as a repository of information to exploiting it more fully to help inform and set the direction of investigative work. The use of more sophisticated querying and database technology has enabled patterns of criminal activity to be identified that can be used for profiling. Criminal profiling characterises the behaviour of an unknown suspect by looking for common patterns within crimes that reflect the behavioural characteristics of the offender [16]. The two central features of criminal profiling are the MO (modus operandi - the particular methods employed by a given criminal, which may change as the criminal becomes more practised) and what is known as the "signature" (the reasons the offender undertakes the crime). The technique relies on the assumption that two criminals rarely share the same MO and signature.

Another form of profiling is based on the assumption that the locations of serial crimes are not completely random, but conform to an underlying pattern which can give clues about the offender. Geographic profiling uses information about a series of crimes to generate a profile, generally output as a colour-coded map, which shows the most probable locations of the centre of activity. Such central points generally correlate with offenders' home addresses or places of work. Profiling approaches and techniques can also be used to detect more general patterns of criminal activity such as areas which have high crime rates or particular sets of circumstances or features which make crime more likely. Such high-level profiling can be used to inform policing strategy or crime prevention measures.

Terrorism and some forms of criminal activity are generally perpetrated by groups or organisations rather than individuals. Strategic crime, the combination of organised crime, such as drug trafficking, and terrorism, of a quality and quantity that threatens national security is on the increase as terrorists look for ways to fund their activities [15, 17]. In investigating and countering such activities there is a requirement to analyse associations between individuals and groups, for example personal relationships, communications and financial transactions, to reveal patterns in what might otherwise appear to be unconnected data. These requirements have led to a range of techniques being developed many of which share the approach of representing people, events, organisations and so forth in a network. Techniques include the use of charting conventions (such as those developed by Anacapa Sciences Inc. of Santa Barbara, California [2]), social network analysis, and link analysis. [21].

A number of charting conventions have been developed and put into use [20]. One of these, time event charting can be used to present and analyse the activity of a group chronologically. Symbols are used on the charts to represent incidents or important events and contain a short description and a date stamp. These are connected using lines representing dependency and the flow of time. Another charting technique, flow analysis can be used to investigate known criminal communities. Flow graphs may be constructed to model the flow of commodities, such as finance or weapons, between members of a community. Relationships between events and/or members can be derived leading to patterns of criminal activity being identified.

Social network analysis has long been established as a research area in social science. Historically it has involved studying small social groups by representing the "actors" (usually people) as nodes in a network and various associations such as family or friendship ties, economic flows, behavioural connections, and so forth as links between the nodes. This research has led to the development of a range of algorithms to assign attributes, such as importance or prestige, to actors according to their position in a network. Other algorithms have also been developed to assign attributes to the network as a whole or to sub-networks, such as structural balance or determining cohesive subgroups of actors within the network [24].

Social network analysis is increasingly being used to map and analyse criminal and terrorist networks, such as the one responsible for the September 11th, 2001 attacks, revealing useful insights into the methods and dynamics of these organisations [14]. Information gained through this analysis has the potential for use in countering such groups. It can, for instance, reveal the actor(s) whose removal from a group would cause the greatest destabilisation to that group. A network can be hierarchical in structure or distributed and each will respond differently to attempts to destabilise it, the removal of what might appear to be the most obvious actor, a leader for example, may not have the greatest destabilising effect. This problem may be exacerbated if critical intelligence is missing or if the network is very large. Tools and techniques need to be developed to cope with such problems within complex networks [5].

Closely related to social network analysis is what we shall call "Link Analysis". Link analysis shares several features with social network analysis in that actors are represented as a network of nodes connected by associations such as kinship or common business interests. However link analysis can be seen as a technique which has grown out of the requirements of law enforcement bodies to represent and understand the structure of groups they investigate. Therefore it has a different emphasis in that diagrams may contain a much greater diversity of object types and links. Link analysis has not developed as an academic field and consequently does not have the rich underpinning theory of social network analysis. Nevertheless the approach has been used as a basis for software tools which help law enforcement agencies represent and visualise criminal networks.

There is significant potential for the use of these more advanced network analysis techniques in countering criminal and terrorist activities. The underlying record-based structure of the relational databases currently employed however does not easily support this type of analysis. Due to the covert nature of criminal and terrorist activities, intelligence is scarce and will be spread across a number of sources with each piece of information only becoming significant when combined with other data. Multiple organisations and agencies collect this information using different processes, storing it in multiple formats within disparate databases, resulting in a problem of data interoperability that will need to be addressed if effective sharing and analysis is to take place.

3.0 ISSUES IN APPLYING ANALYSIS TECHNIQUES

Data that underpins the analysis techniques discussed in the previous section is likely to originate from standard data collection processes, such as crime scene reporting and subsequent investigative work. Such data is held in a variety of manual and electronic databases that must be accessed or queried by the analyst in support of the technique being employed. Unfortunately, evidence relevant to a particular investigation may be stored in databases held by a variety of organisations. With counter terrorism there is the additional problem that intelligence analysis may make use of data originating from general surveillance or even news monitoring thus further widening the kinds of input that must be accommodated.

Therefore, the collection and the subsequent sharing of available data are two crucial elements in the use of intelligence. Krebs [14] highlights the importance of data sharing in predicting possible dangers, commenting that information relating to the September 11th attacks was held by various agencies or countries and that had it been shared it would have proved most useful for the identification of suspects. Information fusion*,* defined as “the use of computer technology to acquire data from many sources, integrate these data into usable and accessible forms and interpret them” is now seen as a key technical problem for intelligence analysis [18]..

Currently, there is a tendency for each intelligence analysis technique to be supported by standalone tools. Examples of such tools include ECRI’s *RIGEL profiler* [9] for geographic profiling and i2 Ltd’s *Analyst’s Notebook* [12] for link analysis. Generally, each tool supports one technique and some form of data pre-processing is required before the data is in a form appropriate for the technique.

The databases supporting each tool remain independent. Should investigators need to carry out standard database queries as well as looking for an association between individuals they would require access to distinct tools. This operational problem can add to the cost and slow the tempo of an investigation. The problems arising from the requirement for data pre-processing are better addressed by re-examining the data models that underpin the database. These data modelling issues are discussed in the next section.

4.0 DATA MODELLING ISSUES – RECORD-BASED VERSUS GRAPH-BASED MODELS

Most database systems use aggregation structures, or records, as the basic way of organising information. If, for example, a system is to hold information about people then the particular data items to be stored, such as, name, address, sex, date of birth, will be grouped together into what is known as a record structure. Information on any person will be stored using this same record structure or template. Records of the same type are stored together. Relational database systems, based on the so-called relational model of data [7] are perhaps the most common record-based systems. The relational model of data is essentially a record-oriented data model though it uses a slightly different terminology – a record structure specifies a table that is used to store groups of related facts. A relational table holding information on people might look something like that shown in Figure 2 (note that the title row would not be stored as part of the table). The relational model uses the term “row” rather than record to underline the point that the underlying implementation may or may not store data in the same form in which it is presented to a user.

Name	Address	Sex	dob
John Smith	21 London Road, Reading	M	28/Jun/67
Mary Jones	Flat 4, Victoria Gardens, Didcot	F	12//Sep/81
Bill Adams	34 Muswell Road, Abingdon	M	3/May/78
. . .	. . .	. . .	. . .

Figure 2: Example of a Relational Table

Aggregation structures (known as records or tables according to the system being used) are the predominant organisation which underpins most current database systems. However there are other ways of organising information and Kent [13] has underlined the distinction between record-based and graph-based organisations, both of which can be used as the underlying representation of information for databases. The disadvantages of record-based systems will be explored before discussing the particular characteristics and advantages of graph-based databases.

4.1 Record-Based Data Models and Intelligence Data

A record-based database will use a variety of different record formats according to the application it is supporting. For example a database of criminal intelligence information would probably have different record formats for people, vehicles, locations, incidents, contact reports and so forth.

The need for an appropriate set of record (or table) structures to be defined prior to loading the database gives rise to the requirement for a database design phase. This is needed since it is not always clear what constitutes an appropriate set of record or table designs to support a given application. A number of design methods have been developed to address this problem, most notably entity-relationship modelling [6], and the issues involved in determining a reasonable design for a given application area are well understood. However these methods rely on being able to start from, or at least elucidate, a clear set of requirements in terms of the information to be held and its characteristics. This is not problematical for most standard applications but it is much less clear that one can define the requirements for a database to hold intelligence data given that it will not always be possible to foresee the kinds of information that may need to be stored.

Besides requiring some initial design, record-based systems implicitly assume that data is largely homogeneous. For example a company's customer database will hold similar information for each customer (name, address, contact details, credit rating etc). However a database to support intelligence analysis has to deal with data which will be fragmentary and non homogeneous. The information that may need to be held on individuals is diverse and may include a large range of items such as name, aliases, date of birth, address or addresses, place (or places) of work, frequently visited locations, associates, distinguishing marks and so on. However for any individual recorded in the database only a few of these pieces of information are likely to be relevant or known. Thus, in using record structures or tables, the database is likely to appear very sparse with many missing items of information.

Finally, record-oriented systems tend to obscure the network or interconnected nature of data. This can be seen by considering the example database shown in Figure 3 where tables hold information on people, incidents, and involvement in incidents (people and incidents have been given a special identifier). In analysing such data it is often useful to look for associations, direct or indirect, between people. In this case any search software would probably be able to find links between people by virtue of their having been involved in the same incident - for instance the people represented as P008 and P128 have a connection through both having been involved in incident Inc003. There is also an association between these people and person P001 by virtue of their having been involved in an incident at the latter's address. However this association is not so obvious since the corresponding table columns have different names (address and location). If we are to automate the process of uncovering such connections then determining the ways in which records or tables may be associated with each other becomes particularly problematic.

The software would need to be parameterised to take into account possible matches between address and location (for example) but not between address and description. Such complications make it very difficult to develop general purpose software to find associations in databases.

Person

P_Id	Name	Address	Sex	DoB
P001	John Smith	21 London Road, Reading	M	28/Jun/67
P005	Mary Jones	Flat 4, Victoria Gardens, Didcot	F	12//Sep/81
P008	Bill Adams	34 Muswell Road, Abingdon	M	3/May/78
. . .	. . .	. . .	. . .	. . .

Incident

I_Id	Date	Time	Type	Location	description
Inc098	12/Jan/04	02.45	meeting	4 Beech Crescent, Walcot	. . .
Inc003	23/May/04	19.10	meeting	21 London Road, Reading	. . .
Inc031	5/Jun/04	23.20	argument	Kings Head Pub, Slough	. . .
. . .				. . .	. . .

Involvement

Incident	Person
Inc003	P128
Inc003	P008
Inc098	P004
Inc098	P321
. . .	. . .

Figure 3: Example Tables (Person, Incident and Involvement) from an Intelligence Database

A further problem arises since none of the query languages associated with record-based systems (such as SQL for relational systems) are even capable of expressing queries of the form "*How are A and B connected?*" Nor is it clear in such systems how the answer should be presented since it may involve a relatively complex chain of matches.

4.2 Graph-Based Data Models and their Advantages

Graph-based data models, such as the binary relational model [1], have two main characteristics:

- Entities of interest in the application domain are modelled by special identifiers (sometimes known as "surrogates") which stand in a one-to-one relationship with those entities.
- All information in the database is represented by simple named connections (binary relations) that link a surrogate either to a value (such as a number or character string) or to another surrogate.

For example, a database to hold criminal intelligence data might use a number of special classes of surrogate, such as person, location, or incident. Information about instances of these can be held by binary relations (simple associations) such as `name` (from a person to their name), `address` (from a location to its address), or `associate` (from a person to his or her associates).

Consequently a graph (or binary-relational) database represents information in a form that is directly analogous to a directed graph, such as the one shown in Figure 4. Here we have followed a convention of using solid dots to represent entities such as people or locations. Each entity has been given a unique identifier beginning with a capital letter, such as P001 or Inc003. Text representing information has been enclosed in quotes to distinguish it from the entity identifiers.

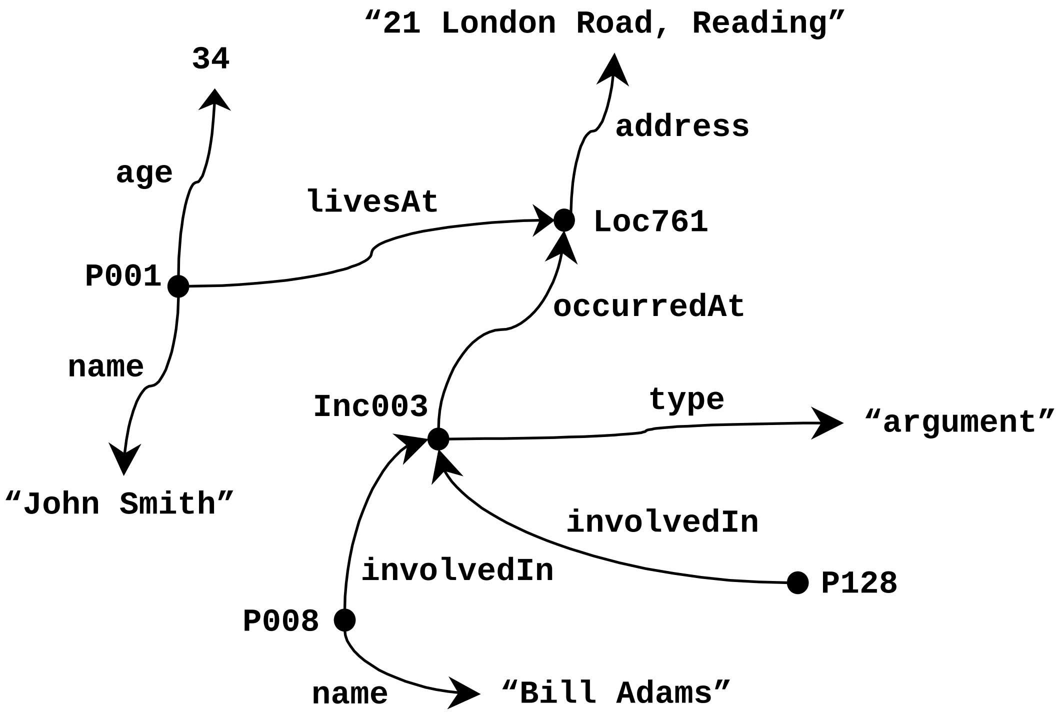


Figure 4: A Fragment from a Graph Database

There are a number of features of this representation of data which make it well-suited to applications such as intelligence analysis:

- The database can be set up without the need for detailed design – the main question is what entity types (such as people or incidents) are of interest in the application domain. Relationships (such as involvedIn or name) can be added as and when the need for them is perceived.
- The representation copes well in situations where the data is sparse – if a piece of information is not known then it is simply not held in the database.
- The model explicitly captures some idea of entity identity since real-world entities are directly modelled by a corresponding special identifier in the database. This means that the existence of distinct entities can be modelled even if we only have partial information on the entity. (In the diagram a person entity P128 is recorded as existing even though we have virtually no information about the person other than that they were involved in a particular incident).
- Queries concerned with asking questions such as *"What is the association between X and Y?*" can simply be treated as searches for paths through the database graph.
- The database lends itself to visual representation as a network or link diagram. Although it is not practicable to present an entire database of any size in this form, sub-portions of interest can still be displayed graphically (this aspect will be discussed further below).

The underlying representation of a graph database can be very simple. One approach uses what is known as a triple store [11] – a large table or repository of triples where each triple corresponds to one arc in the database graph. For instance the database segment shown in Figure 4 could be represented by a store of the form shown in Figure 5. The advantage of such a store is that new kinds of information can be recorded simply by using a new association in the middle column.

P001	name	“John Smith”
P001	livesAt	Loc761
Loc761	address	“23 London Road, Reading”
Inc003	occurredAt	Loc761
Inc003	type	“argument”
P126	involvedIn	Inc003
P008	involvedIn	Inc003
P001	age	34
P008	name	“Bill Adams”
. . .	. . .	. . .

Figure 5: Portion of a Triple Store for Graph Database of Figure 4

Graph data models and triple stores as discussed above are not however ideal for all applications. When largely homogeneous data is being processed they may not give as good performance when compared with relational databases and they are ill-suited to forms of data that are not naturally seen as relating to attributes of real-world entities. For example a set of sensor readings or performance data may be complicated by the need to define entity identifiers that do not have an intuitive real-world correspondence.

However we believe that graph-based representations have many advantages in application areas such as intelligence analysis and below a system is discussed that has been implemented over such a data model. The particular advantage that it offers is the facility to search the database for associations without losing any of the query power of conventional systems.

5.0 HYDRA: A SYSTEM BUILT ON A GRAPH-BASED DATA MODEL

An example of a database system that has been developed over a graph-based data model is Hydra [3]. The approach adopted by Hydra is to integrate database and associated query facilities in a so-called functional programming language [10, 19]. Functional languages support a style of programming which abstracts away from the details of algorithm execution to a far greater extent than do ordinary programming languages such as C or Java. This allows programs to be expressed in a very succinct form. The novel feature of the Hydra system is that it integrates the facilities required to search for associations in a database with ordinary query facilities.

5.1 Setting Up a Database in Hydra

There are two kinds of values that can appear in a Hydra database. These directly mirror the distinction made above between ordinary values and surrogates:

- Ordinary values such as numbers or text segments (known as strings) which can be manipulated using standard operators (addition, subtraction etc), and
- Special purpose values that are used to represent application entities. These are grouped into classes of entity, such as person or car for instance.

Setting up a Hydra database involves first specifying some entity classes to be used. For example:

```
entity person, location;
```

defines the classes `person` and `location`. Individual instances of person or location can now be added by giving them a unique identifier that must begin with a capital letter, hence the statements:

```
person ::+ P001, P008;
location ::+ Loc761;
```

introduce three entity identifiers or surrogates, two persons and a location. Associations must be declared before particular instance-level links are introduced. For example to introduce an age association the user would enter the statement:

```
age :: person -> int;
```

specifying that a person can be associated with an integer (whole number) age. To define the age of a particular person we can now enter:

```
age P001 = 34;
```

which states that the age of the person with database identifier `P001` is 34. Associations can also link entity instances, hence the association `livesAt` declared as follows:

```
livesAt :: person -> location;
```

can be used to link a person to their principal residence, as for example:

```
livesAt P001 = Loc761;
```

Hydra distinguishes between single-valued associations such as `age` (a person cannot have more than one age) and associations which may link a given entity to several values or entities. For example an association `frequents` may be used to link people to the locations they frequent. Such multi-valued associations can be declared as follows:

```
frequents :: person -> [location];
```

where the square brackets enclosing location signify that one or more locations may be associated with a given person. Thus if we introduce two more locations:

```
location ::+ Loc100, Loc101;
```

then we can specify that an individual frequents these locations as follows:

```
frequents P008 =+ Loc100, Loc101;
```

The combined effect of the Hydra statements given is to construct the database graph in Figure 6.

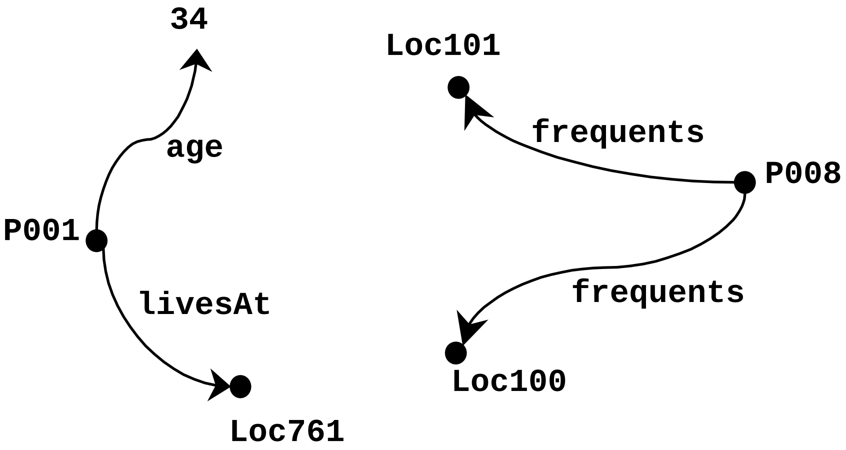


Figure 6: Database Graph Constructed by Hydra Statements

At any stage new entity types, associations, and entity instances or links can be added so that the database graph can continually change as new entity types or new kinds of association are required

5.2 Conventional Query Facilities

Hydra supports all the query facilities that are available in ordinary database systems. To retrieve the age of the person represented by `P001` we can enter

```
age P001;
```

which (given the database graph above) will evaluate to 34. To find the locations frequented by `P008` we enter:

```
frequents P008;
```

and the answer will be returned as a list (lists in Hydra are enclosed in square brackets) of values as follows:

```
[Loc101, Loc100]
```

Besides following associations "forwards" they can also be followed "backwards" by prefixing the name of the link with a tilde. Hence

```
~age 34;
```

finds those people who are recorded as having age 34. Given the database above it will evaluate to:

```
[P001]
```

Note that all links, when followed backwards, return lists of results since there can be no guarantee that any single-valued link, when reversed, remains single-valued.

The previous examples have involved retrieving information about a given entity or those entities having a particular value for some associated attribute. We can also carry out queries that involve scanning through all the entities in a class. To do this the primitive `like` must be used. For example the expression:

```
      like ?person;
```

returns the list [P001, P008] of all the person entities currently defined. This can be combined with a special facility of functional languages, known as "list comprehensions" [4], which is supported by Hydra to allow more complex queries. For example the following query (executed against a larger database than the one built up so far):

```
      [(name p, age p, address(livesAt p)) | p <- like ?person];
```

lists the name, age and address of all the people defined in the database. It will return a result of the form:

```
     [("John Smith", 29, "3 King Street, Windsor"),
      ("Mary Jones", 39, ?string),
      ("Mike Adams", ?int, ?string), . . .]
```

where values such as ?int or ?string correspond to unknown integer or string values where the corresponding link was not defined in the database. More complex queries can easily be built up so:

```
      [(name p, address(livesAt p)) | p <- like ?person
                                    | age p > 20
                                    | frequents p != [] ];
```

returns the name and address of everyone whose age is greater than 20 and who is known to frequent at least one location (i.e. the list of places they frequent is not the empty list []).

Hydra also supports the facility to define general purpose functions (known as secondary functions) which do not form part of the database graph. Hence we can define a function length which when applied to any list will return the length of the list. Thus

```
      length [1,2,3];
```

evaluates to 3 and

```
      length [];
```

evaluates to 0. General purpose functions like this can be combined with database queries so that:

```
      length [p | p <- like ?person | age p < 50 ];
```

will return the number of people aged under 50 who are recorded in the database.

Using a combination of simple retrievals, list comprehensions, and secondary functions it is possible to code up any query that can be supported by a conventional database system. Thus Hydra provides all the retrieval power available in other systems, such as relational databases for example, even though it is built over a different representation of data.

5.3 Supporting Link Analysis

A particular aspect of the examples given above is that the associations to be used in evaluating the query are known in advance. In most database systems it is not even possible to formulate a query unless one knows which tables (or record types) or fields are required to evaluate the query. The particular feature of Hydra is that it is possible to formulate queries without knowing what associations must be used to evaluate the query. Hydra provides this facility by exploiting the network or graph organisation of the data and treating some queries as simple searches for paths through the database graph. These path-searching facilities are provided by a number of special primitives that are built into the language.

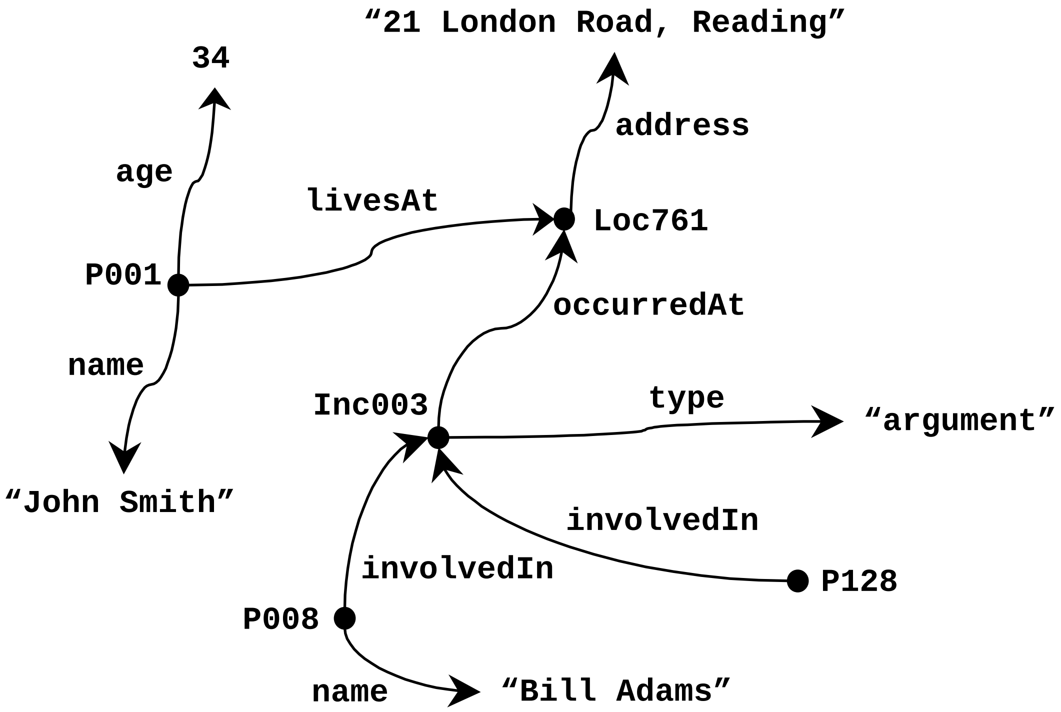


Figure 7: Example Database Graph

The Hydra primitives `to` and `from` return respectively all the links that may come from a value or entity of a particular type and all those that may go to an entity or value of a particular type. Hence given the state of the database shown in Figure 7, the query:

```
from Loc761;
```

returns a list such as:

```
[address, . . .]
```

containing all the links which might be defined for a location entity (note that other link types for locations may have been declared but not defined for `Loc761`). The query:

```
to Loc761;
```

returns a list such as:

```
[~livesAt, ~occurredAt, . . .]
```

of links which, if traversed backwards, are potentially defined for a location entity. Using primitives such as this it is possible to define functions which will find everything known about an entity instance (i.e. follow all the arcs connected to the entity) so that we can enter a query such as:

```
known Loc761;
```

and receive a result:

```
[(address, “21 London Road, Reading”),
     (~livesAt, [P001]), (~occurredAt, [Inc003]) ]
```

The precise definition of a function such as `known` is not important but it is built up using the `from` and `to` primitives. Whatever the state of the database it can find everything connected to a node in the database graph. Similar functions to start at any node and find all the values or entities within a given distance (say 2 links) can also be defined and executed.

Another facility, corresponding to the question “*How are X and Y associated with each other in the database?*” can be answered by the `link` primitive. For example the query:

```
link 4 P128 “John Smith”;
```

will find any connection between `P128` and the text string “`John Smith`”. which involves traversing 4 links or less. Given the database graph shown in Figure 7 the result will be a list of alternating nodes and arcs of the form:

```
[[P128, involvedIn, Inc003, occurredAt, Loc761,
                ~livesAt, P001, name, “John Smith”]]
```

Even more complex queries can be built up which treat the database as a large network and look for connections. For example we can, in Hydra, develop a function `centre` which takes a search depth and a list of start nodes (i.e. database entities or atomic values such as numbers or strings) and searches for database nodes which have a connection of within the search depth to each of the start nodes. Such a query corresponds to a simple question such as “*There is a rumour that Bill Adams had an argument with someone in which John Smith was mentioned*”. A corresponding Hydra query would be:

```
centre 3 [“John Smith”, “Bill Adams”, “argument”];
```

which will return the result:

```
[Loc761, Inc003]
```

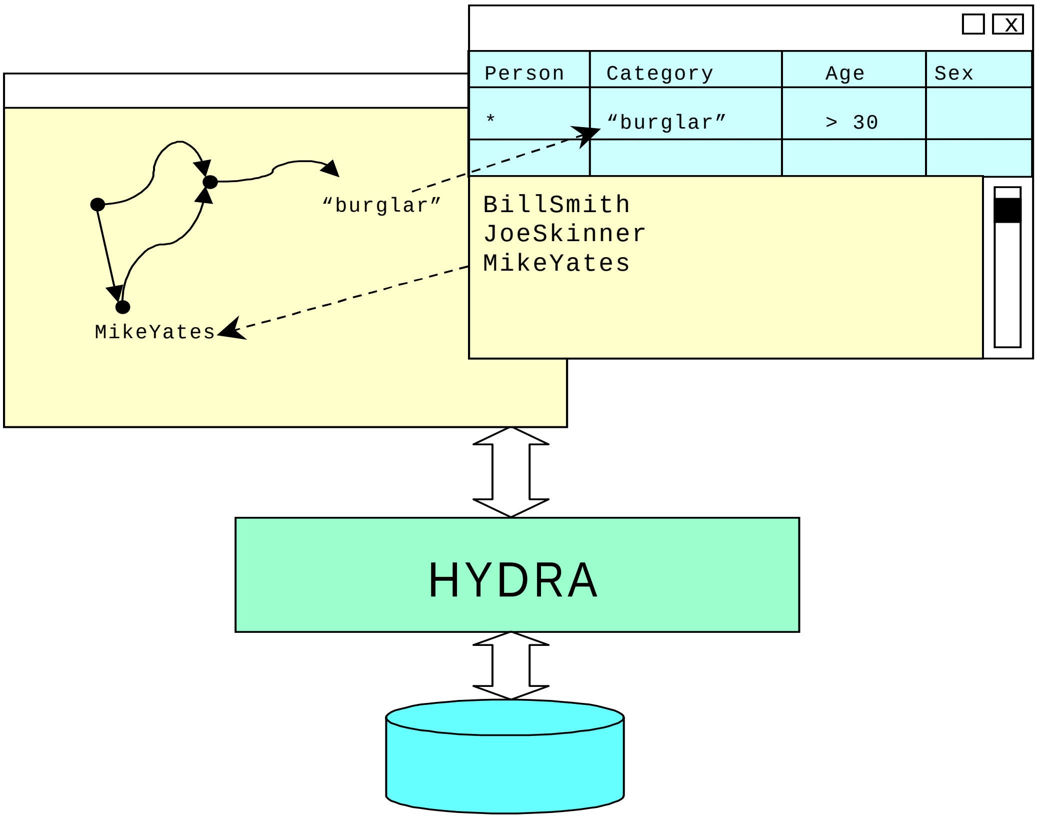
since both of those values have a path through the database of not longer than 3 links to each of the values specified in the initial list (further details on how such a facility is defined are given in [3]).

The unstructured queries outlined above simply cannot be carried out in most conventional databases. Their inclusion in Hydra represents, we believe, a powerful facility that is particularly appropriate to domains such as intelligence analysis. Furthermore since Hydra is a full programming language it is relatively simple for an experienced user to develop new searches for particular kinds of associational patterns or features in the database graph. Thus particular social network algorithms or link analysis features can be implemented as and when required.

5.4 Graphical Interface Possibilities

The potential for data visualisation of graph databases is not exploited in the Hydra system which has a textual interface. However it is quite feasible to use it as the basis for an interface that would allow queries to be specified either graphically (on a canvas) or in terms of a spreadsheet style format. Results could then be presented graphically, as a database sub-graph, or in tabular form as appropriate:

The style of interface feasible is shown in Figure 8. Besides insulating ordinary uses from the complexities of a textual interface it has the further advantage that multimedia information, such as images, can easily be accommodated on a canvas.



Person	Category	Age	Sex
*	"burglar"	> 30	

BillSmith
JoeSkinner
MikeYates

Figure 8: Possible Hydra Interface

6.0 CONCLUSIONS

This paper has discussed some of the advantages of graph-based data models. Such models have a number of useful characteristics for application domains where information about human and real-world activities and relationships needs to be captured. One particular advantage of such models is that they are very close to the form of representation that is used in application areas such as social network analysis and link analysis. Consequently a graph database, or a portion of a graph database, can easily be displayed using notational conventions that are similar to those used in link analysis.

In discussing the benefits of graph data models we have used one system, Hydra, as an example to demonstrate that many of the benefits of graph databases have in fact been realised in a working prototype. Hydra exploits a graph-oriented representation of information to provide a more powerful range of query facilities than are available in most systems. Graph-based databases could also be integrated with other systems and programming paradigms. In particular an environment such as Java, which is multi-platform and which provides a rich and extensive set of libraries, would provide a powerful framework into which a graph-based system could be integrated. This would then benefit from a rich set of graphical user interface facilities that could be used to present the data in a variety of ways.

Furthermore, although Hydra is built on top of its own special purpose storage manager there is no reason why conventional systems could not be used to support graph databases. What this means is that graph-oriented representations can easily be incorporated into the design and implementation of systems that are concerned with storage and manipulation of intelligence data.

There are other issues that remain unresolved in the storage and exploitation of intelligence in a system such as Hydra. One of these being how to represent levels of certainty and the fact that much of the intelligence gathered may be contradictory or of dubious provenance. Another major issue is the collation and combination of data from a variety of existing repositories that use a range of different database designs.

Use of graph databases does not directly help with such issues though they may form a basis of new systems. In particular we believe that they may provide an approach for collating information from existing databases. Their advantage in this context is that they could support alternative representations of the same information in one database. This would make it possible to automatically transfer existing data into a central repository where techniques, such as link analysis, could be applied.

7.0 REFERENCES

[1] J. R. Abrial, "Data Semantics" in J. W. Klimbie, K. L. Koffeman, editors, Data Base Management, North Holland 1974, pp 1-59.

[2] Anacapa Sciences Incorporated. http://www.anacapasciences.com Accessed July, 2004.

[3] R. Ayres, "The functional data model as the basis for an enriched database query language", Journal of Intelligent Information Systems, Vol 12(2/3), 1999, pp 139-164.

[4] R. Bird, P. Wadler, Introduction to Functional Programming, Prentice Hall, 1988.

[5] K. M. Carley, J. S. Lee, D. Krackhardt. "Destabilizing networks", Connections, Vol 24 (3), 2002, pp 79-92.

[6] P. Chen, "The entity-relationship model: towards a unified view of data", ACM Transactions on Database Systems, Vol 1(1), 1976, pp 9-36.

[7] E. F. Codd, "A relational model for large shared data banks", Communications of the ACM, Vol 13(2), 1970, pp 377-387.

[8] C. J. Date, An Introduction to Database Systems, Seventh Edition, Addison Wesley, 2001.

[9] Environmental Criminology Research Inc. http://www.geographicprofiling.com. Accessed July, 2004.

[10] A. J. Field, P. G. Harrison, Functional Programming, Addison-Wesley, 1988.

[11] R. A. Frost, "Binary relational storage structures", The Computer Journal, Vol 28(5), 1985, pp 359-367.

[12] i2 Limited. http://www.i2.co.uk. Accessed July 2004.

[13] W. Kent. "Limitations of record-based information models", ACM Transactions on Database Systems, Vol 6(1), 1979, pp 107-131.

[14] Krebs, V. E. "Mapping networks of terrorist cells", Connections, Vol 24 (3), 2002, pp 43-52.

[15] T. Makarenko. “A model of terrorist-criminal relationships”, Janes Intelligence Review, September 1, 2003.

[16] J. Mena, Investigative Data Mining for Security and Criminal Detection, Butterworth Heinemann, 2003.

[17] D. Menarchik. “Organizing to Combat 21st Century Terrorism” in The Terrorism Threat and US Government Response: Operational and Organizational Factors. Edited by J. M. Smith and W. C. Thomas. USAF Institute for National Security Studies, March 2001.

[18] National Research Council, Making the Nation Safer: The Role of Science and Technology in Countering Terrorism, The National Academic Press, 2002.

[19] L. C. Paulson, ML for the Working Programmer, 2nd Edition, Cambridge University Press, 1996.

[20] Scottish Police College, Course Prospectus 2004, Criminal Management Division, Criminal Intelligence Analysis Course.

[21] M. K. Sparrow, “The application of network analysis to criminal intelligence: an assessment of the prospects”, Social Networks Vol 13, 1991, pp 251-274.

[22] C. R. Swanson, N. C. Chamelin, L. Territo, Criminal Investigation, Seventh Edition. McGraw Hill, 2000.

[23] D. Tucker, “Combating International Terrorism”, Terrorist threat and U.S. Government Response: Operational and Organizational Factors. USAF Institute for National Security Studies, US Air Force Academy, Colorado, 2001.

[24] S. Wassermann, K. Faust, Social Network Analysis: Methods and Applications, Cambridge UK, Cambridge University Press, 1994.

Database Issues for Intelligence Analysis

Robert Ayres, Ross D Harris, Graham Lee and Steve Smith

Cranfield University

Royal Military College of Science

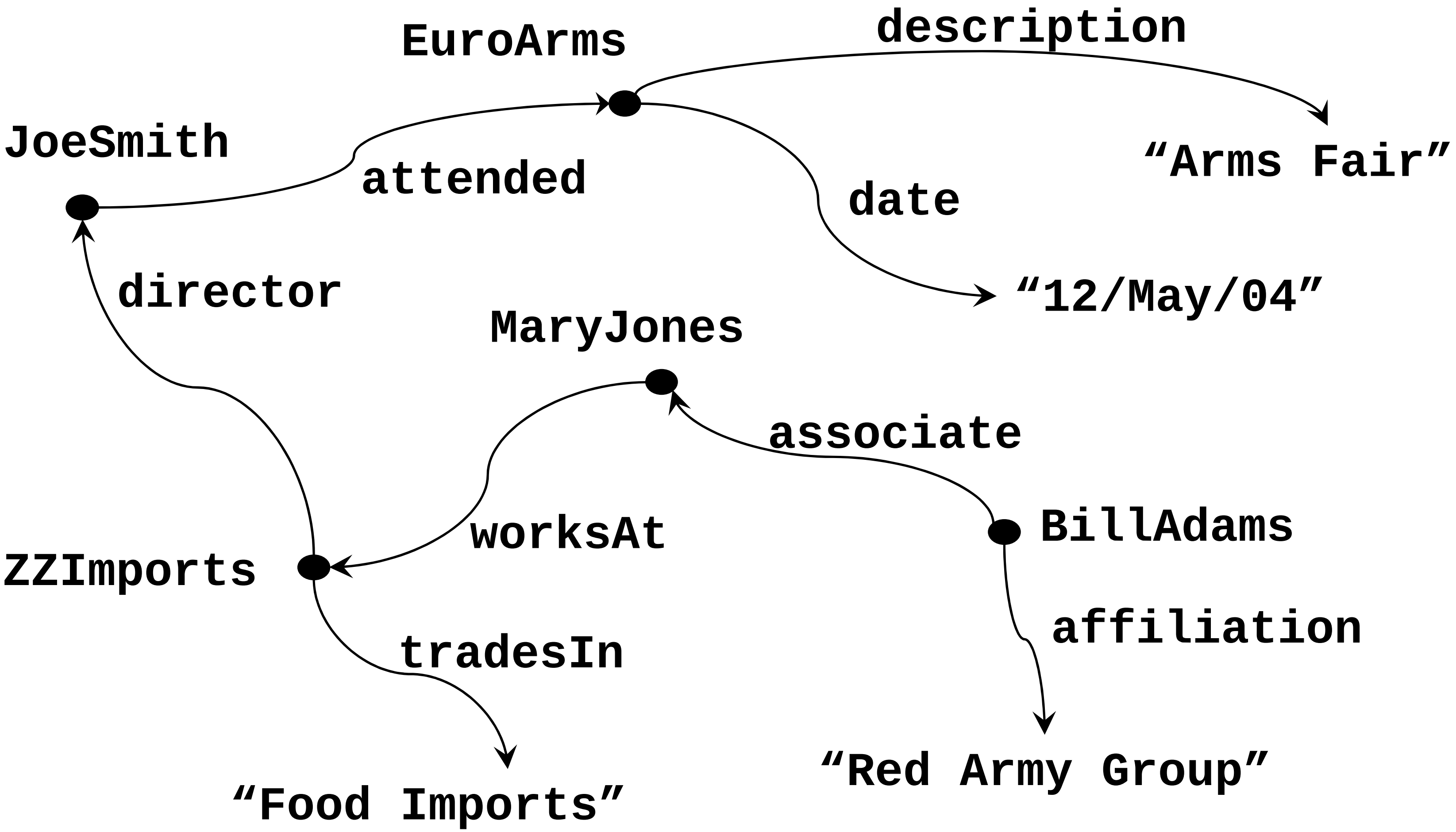
Making Sense of Disparate Information

. . .
Joe Smith is a director of ZZ Imports
Joe Smith attended the Euro Arms Fair on 12/May/04
. . .
. . .
Mary Jones works for ZZ Imports
. . .
Bill Adams knows Mary Jones
Bill Adams belongs to the Red Army Group
. . .

- A key problem in intelligence analysis

Cranfield UNIVERSITY

One Approach - Link Diagrams



Link Diagrams: Issues

- Intuitive presentation of information
  - Can be drawn by software packages
- BUT
  - no support for ordinary queries
  - cannot handle bulk data
  - may not be able to use other techniques
    - eg statistical analysis
- Hence
  - require database technology

Conventional Databases

- Record-based - group (aggregate) closely related facts eg
  - – Person (name, address, age)
  - – Table of person data may look like:

Name	Address	Age
John	12 Summer Street	29
Peter	28 London Road	34
Caroline	12 Summer Street	31
. . .	. . .	. . .

- Database made up of many tables

BUT lose network view of data

Name	Address	Age
John	12 Summer Street	29
Peter	28 London Road	34
Caroline	12 Summer Street	31
. . .	. . .	. . .

Support queries such as . . .

- What is the age of (person with name) John?
- How many people live at 28 London Road?

But not . . .

- What is the association between John and 31?

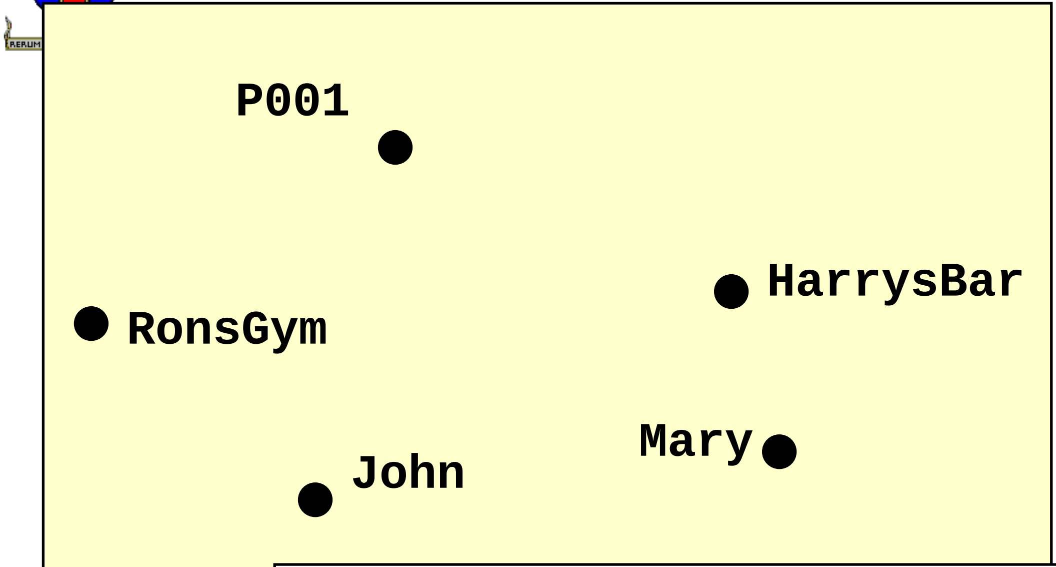
Representation Issues (Kent 79)

- Record-based databases
    - entities represented by groups of facts
- Graph-based databases
    - entities represented by special identifiers
    - facts represented as named links
  - Implications:
    - database becomes a large network
    - very similar representation to link diagrams

Graph-Based Systems: Advantages

- No loss of facilities
- Can also support searches for associations or connections
  - not available in standard databases
  - useful in intelligence analysis
- Obvious visual representation
- Example of a graph-based database
  - Hydra

Hydra - Defining Entities

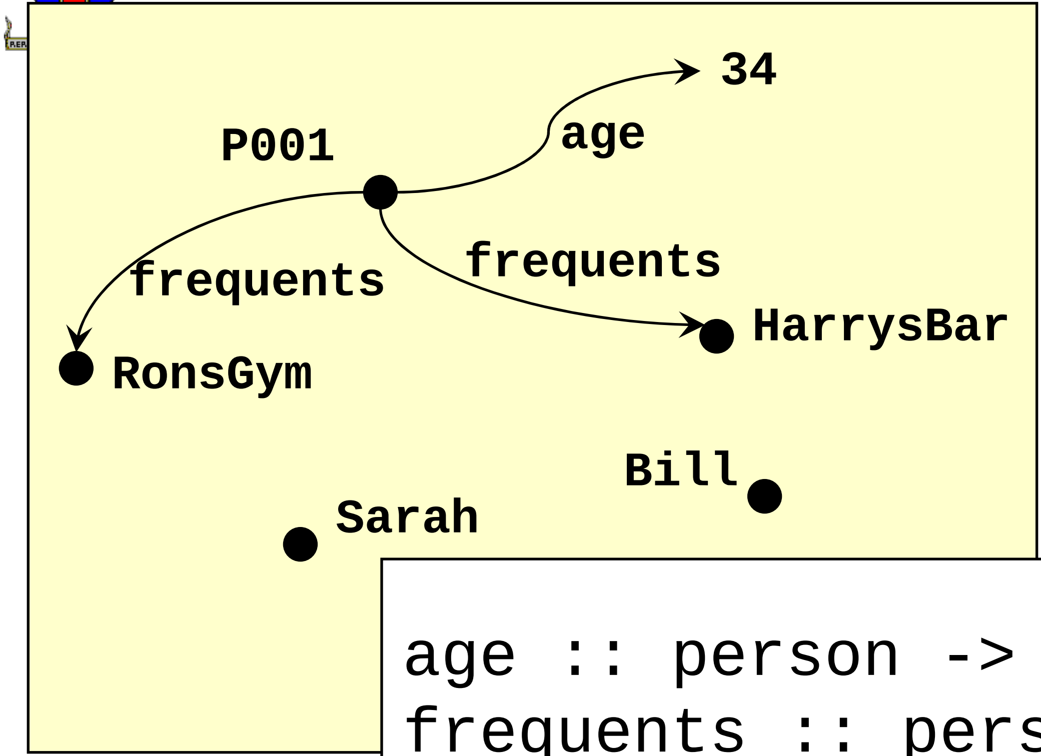


Uses special identifiers to represent real world entities

```
entity person;
entity location;

person ::+ P001;
location ::+ RonsGym, HarrysBar;
person ::+ John, Mary;
```

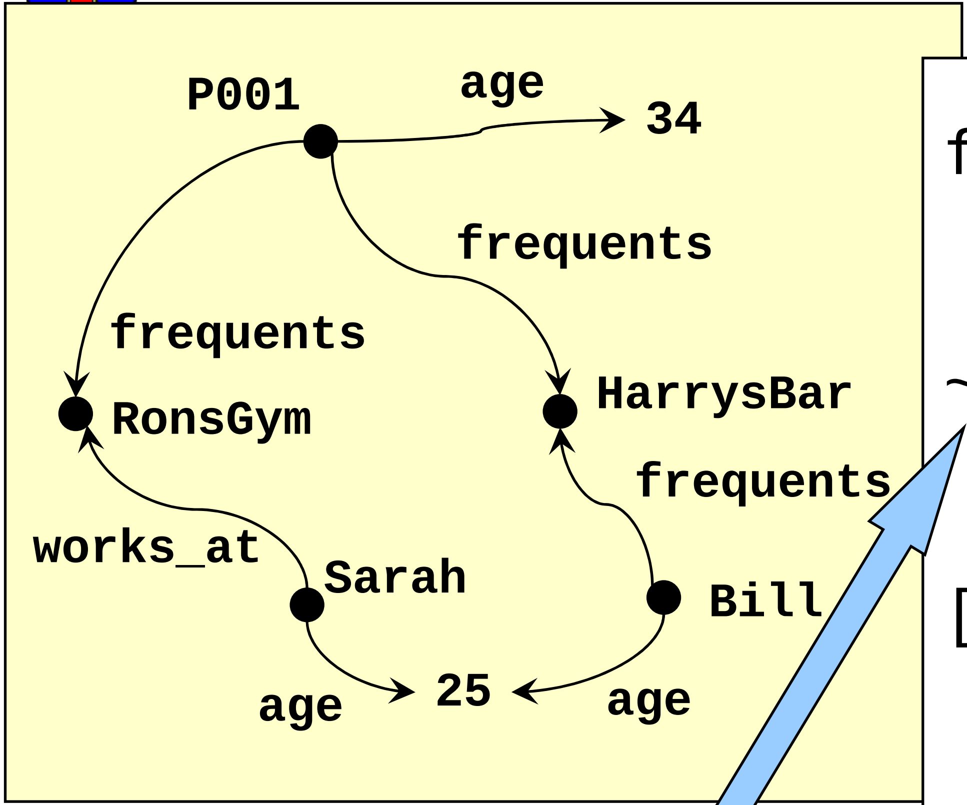
Hydra - Associations



```
age :: person -> number;
frequents :: person -> [location];

age P001 = 34;
frequents P001 =+ RonsGym;
frequents P001 =+ HarrysBar;
```

Hydra: Standard Queries



```
frequents P001;
  --> [RonsGym, HarrysBar]

~age 25;
   --> [Sarah, Bill]

[p | p <- like P001
   | age p > 50];
   --> []
```

~ "backwards" link

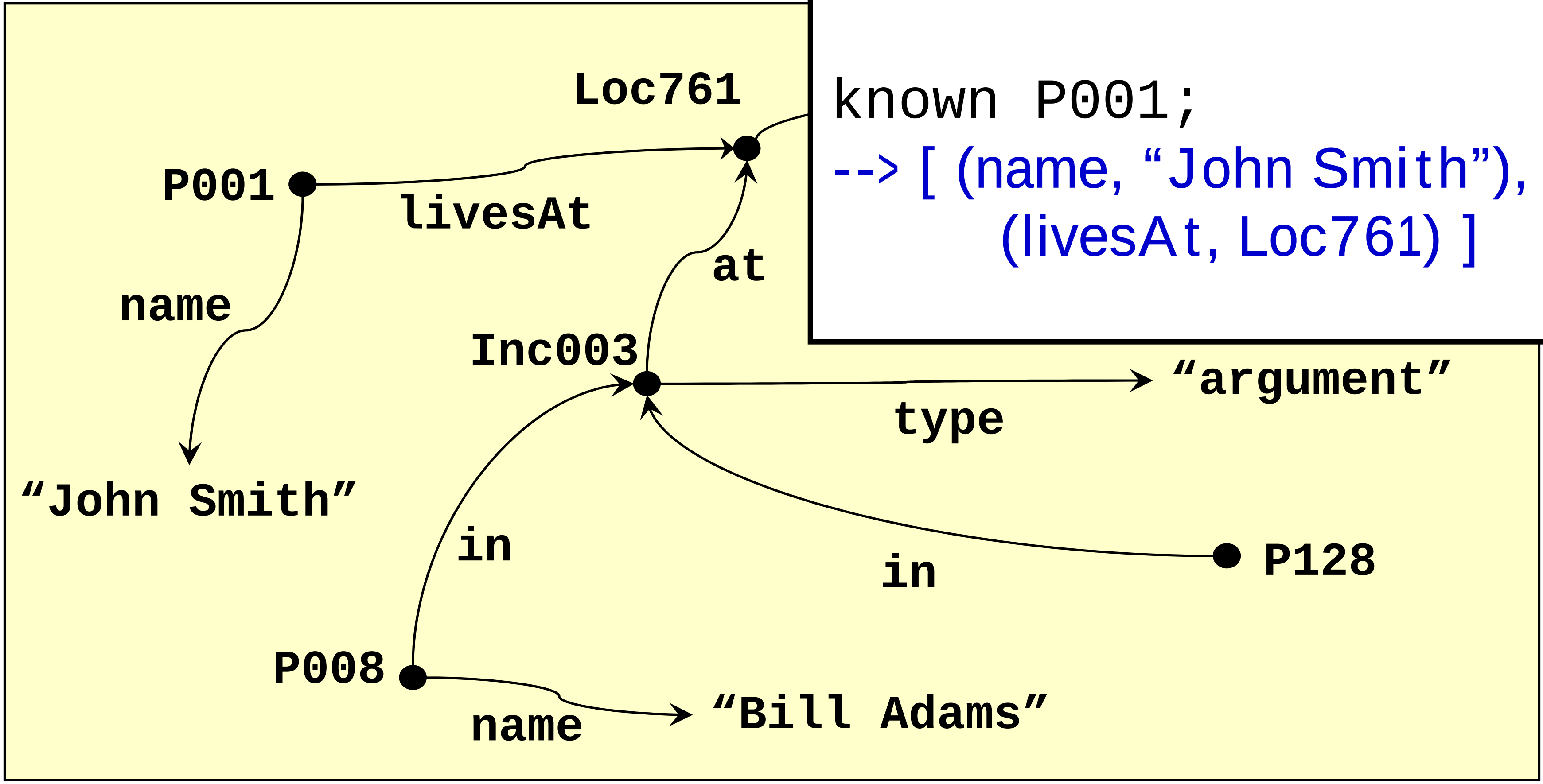
NB: No Loss of Power compared to Record-Based

Finding Connections (1)

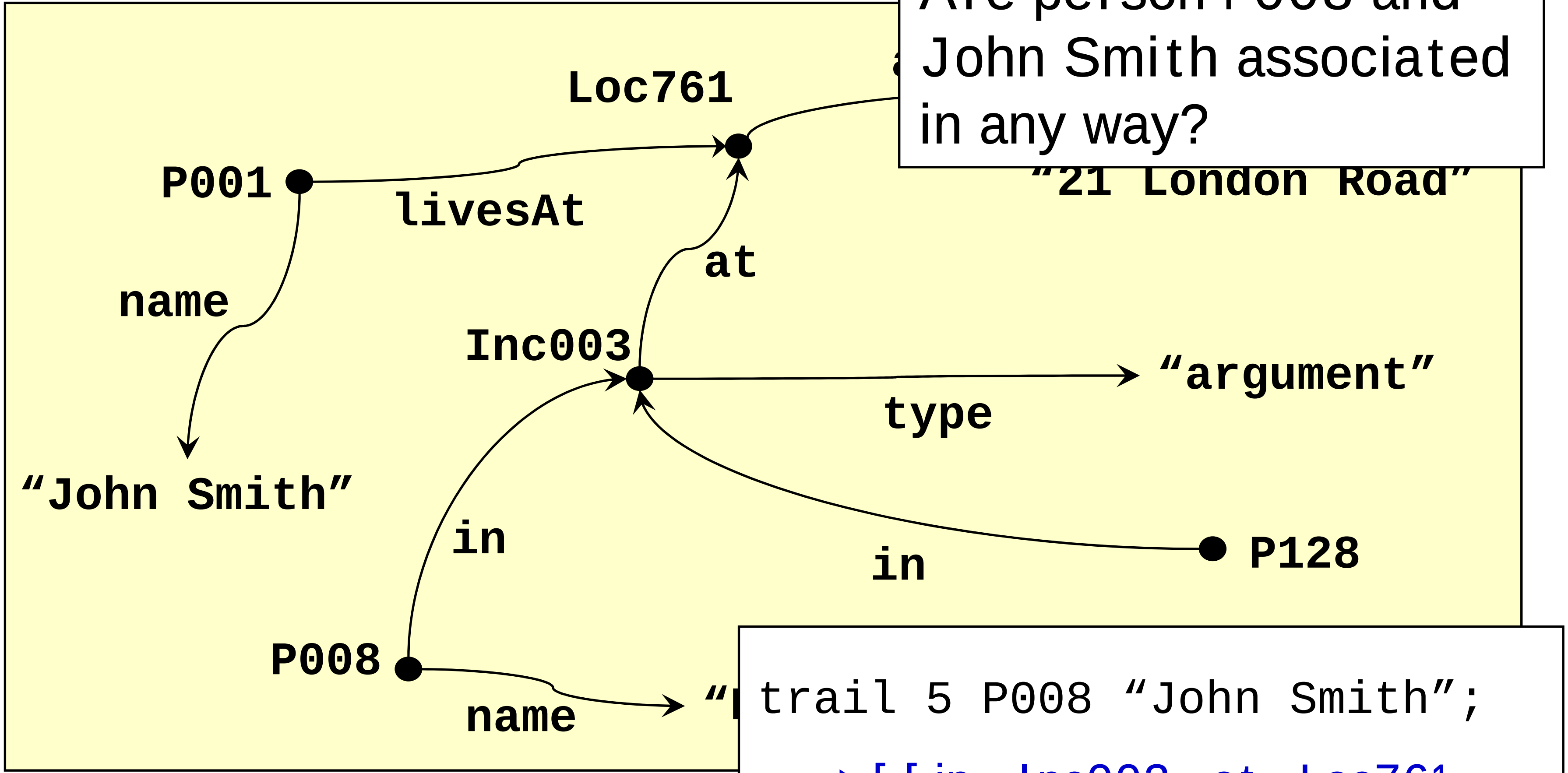
What do we know about person P001?

```
from P001;
  --> [name,  livesAt,  in]

known P001;
--> [ (name, “John Smith”),
        (livesAt, Loc761) ]
```



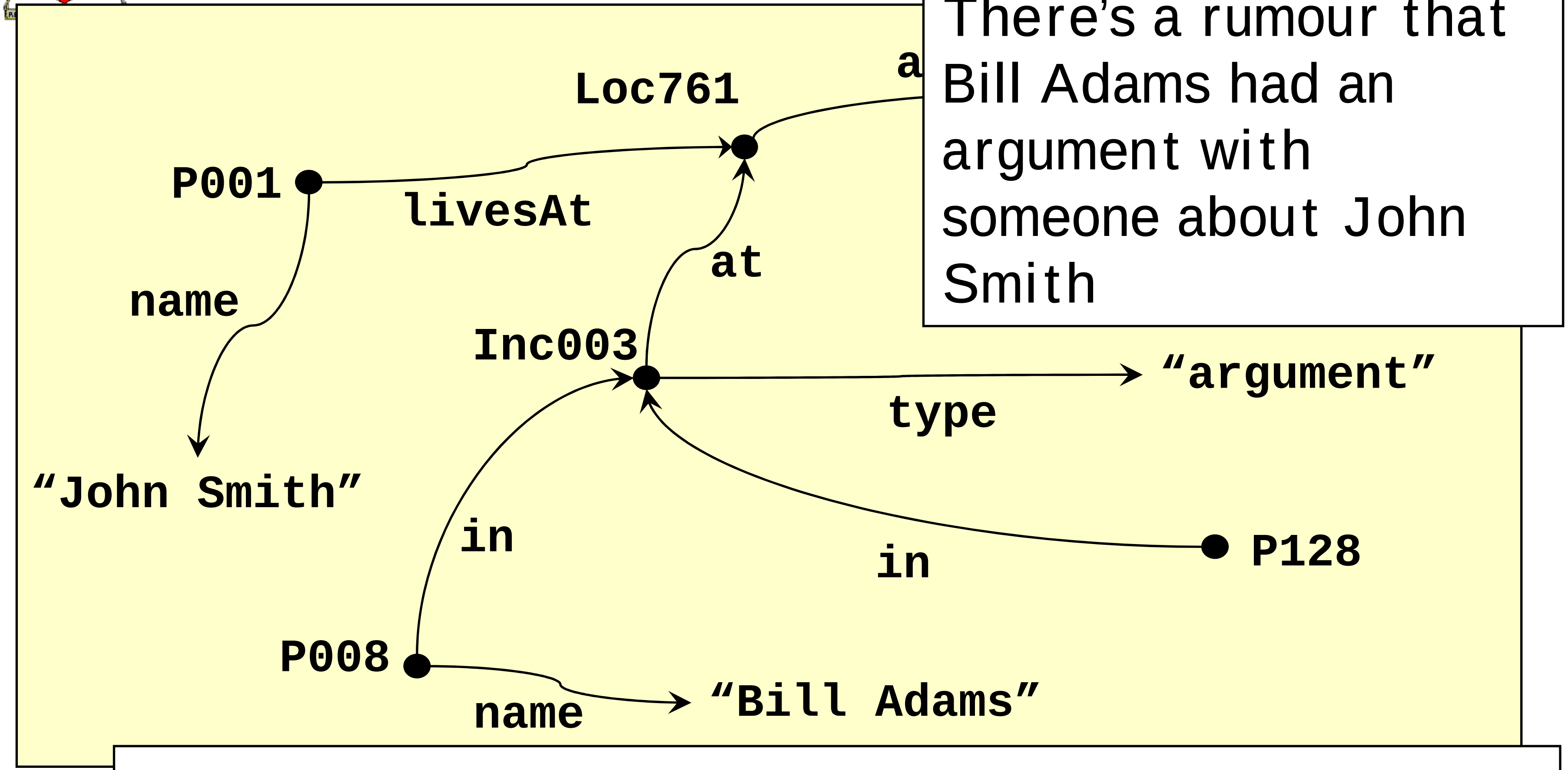
Finding Connections (2)



Are person P008 and John Smith associated in any way?

```
trail 5 P008 “John Smith”;
  --> [ [ in, Inc003, at, Loc761,
          ~livesAt, P001, name ] ]
```

Finding Connections (3)

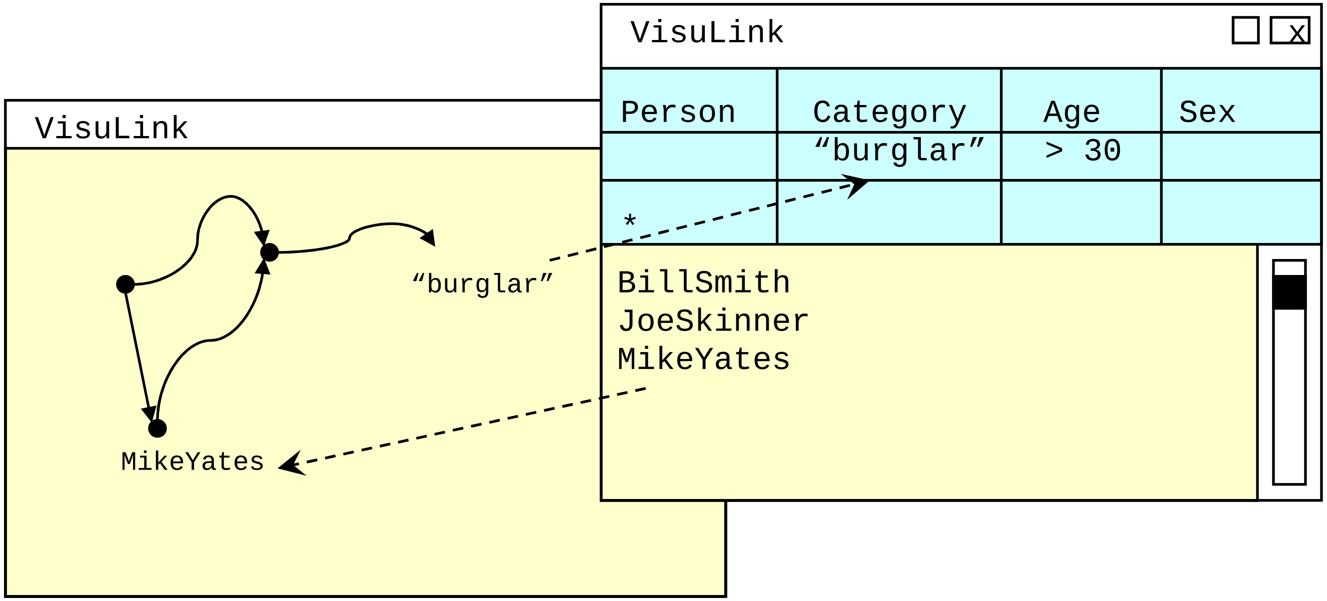


There’s a rumour that Bill Adams had an argument with someone about John Smith

```
centre 3 [“John Smith”, “argument”, “Bill Adams”];
  --> [ Loc761,  Inc003 ]
```

Summary

- Graph-based DBs & intelligence analysis
  - can combine standard facilities & link analysis
- Further Work
  - graphical interface for demonstrator



Person	Category	Age	Sex
	"burglar"	> 30	
*			

BillSmith
JoeSkinner
MikeYates